Computer Vision: (relates to) machine learning, neuroscience, robotics, medical imaging, human-computer interaction, computer graphics

image(video) → Eyes | Camera → Brain | Computer → Understanding

Human Vision: Sensor—eyes Processor—brain (primary visual cortex)
Computer Vision: Sensor—camera, microscopy, medical imaging scanner Processor—computer

Computer RGB channel → color picture

To construct a system of programs, which will divide a picture into regions objects, background areas, chaos.

—ImageNet Challenge:
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 1.2million images, 1000 classes
—Classification Challenge
—Detection Challenge:
Intersection over Union $IoU = \frac{A \cap B}{A \cup B}$ detection is correct if IoU>0.5

—Image Classification, what is in the picture?
—Object Detection, where are they roughly in the picture?
—Image Segmentation, draw each object accurately.

Application:
—face detection,
Haar features(binary filter, calculate density difference between eyes & rest)
—automatic number plate recognition,
—google street number,
—autonomous driving,
—XBOX 360 Kinect, HCI, RGB camera+depth camera
depth image→body parts→key points→skeleton
—face re-enactment,
—image style transfer,
—drones,
—medical imaging

Image Formation

1.Light
Point Light Source
originates from a single location in space
a small light bulb, the sun remotely
location, intensity and spectrum

Area Light Source

more complicated

ceiling lights in the classroom

2.Reflectance

Bidirectional reflectance distribution function (BRDF) $f_r(\theta_i,\varphi_i,\theta_r,\varphi_r,\lambda)=\frac{dL_r}{dE_i}$

the most general model

describe how much light arriving at an incident direction is emitted in a reflected direction

—Diffuse Reflection

light is scattered uniformly in all direction; BRDF is constant

—Specular Reflection

reflection in a mirror-like fashion; reflection and incident directions are symmetric with respect to the surface normal

For most of the cases, what we see is a combination of diffuse reflection, specular reflection and ambient illumination.

The ambient illumination accounts for general illumination which may be complicated to model,

such as inter-reflection between walls in a room, distant sources (blue sky, sunny outdoor)

3.Computer Graphics (objects → image) Computer Vision (image → objects)

Images synthesized by photorealistic computer games can be used for training computer vision algorithms

It provides detailed semantic annotation for all the images

Complementary to manually annotation, which is time-consuming.

4.Optics

Human Sensors (Retina+ Cone Cell + Rod Cell)

—cone cells responsible for colour vision and function in bright light

trichromatic vision

humans and close related primates usually have three kinds of cones, which have different response curves (S cones respond to short wavelength (blue), M to medium (green) and L to long (red))

occasionally, dichromacy or tetrachromacy

—rod cells play little role in colour vision and function in dim light

rods are more sensitive to light than cones, requires less light to function

rods are primary source of visual information at night

Camera Sensors

thin lens equation 1/f=1/u+1/v

two common types of sensor:

CCD(charged-coupled device), CMOS(complementary metal-oxide semiconductor)

CMOS is used by most smart phone cameras

Sensors convert incoming photons into electron charges and read the values out.

Invented by Bryce Bayer of Kodak

50% green, 25% red, 25% blue (mimic human eyes, which are most sensitive to green light)

Bayer Filter

Only one colour is available at each pixel, the other two colours can be interpolated from neighbouring pixels

By interpolation, at each pixel we can get (R, G, B) values.

Latest sensors for smartphones, 8000H × 6000V pixels (48MP), each pixel size 0.8μm ×0.8μm

5.Colors

Our eyes have 3 types of cone cells

S cell responding to short wavelength light **M** cell responding to medium wavelength light **L** cell responding to long wavelength light

The camera sensors consist of R, G, B sensors.

Most visible colours can be represented by mixing three Primary colours (R, G, B)

CIE 1931 RGB colour space

represent colors on a 2D plane, normalizing the brightness x=X/(X+Y+Z) y=Y/(X+Y+Z) z=1-x-y

X,Y, Z are imaginary primary colours and x, y are chromaticity, colour after removing brightness

sRGB color space (standard RGB space)

Created by HP and Microsoft in 1996 for use on monitors, printers and internet

It is only a subset (gamut) of all the colours that we can see.

The projectors, laptops that you use may not produce all the colours.

Other color space

HSV

Hue (values in [0, 1], colours)

Saturation (values in [0, 1], unsaturated to fully saturated/no white component)

Value (values in [0, 1], brightness)

CMYK (used for printing)

Cyan, Magenta, Yellow, Key (black)

Color Models

Additive (RGB) start from black, add a spectrum of light in mixing

Subtractive (CMYK) start from white (white paper), subtract a spectrum of light (painting)

6.Image Representation: RGB, RGBA or Greyscale

Color representation in computer

①RGB or RGBA (Alpha channel: transparency)

②Grey (Greyscale intensity, R=G=B)

Representation (most common 8 bits, 0 to 255; camera raw files 16 bits)

Quantization Error

floating numbers truncated or rounded

natural images (8 bits, 0 to 255; 16 bits, 0 to 65535)

medical images (12 bits, 0 to 4095)

Quantization signal to noise ratio $SNR = \frac{average\ signal\ power}{average\ noise\ power}$

7.Histogram

histogram divides the intensity range into a number of bins and shows number of pixels or frequencies in each bin

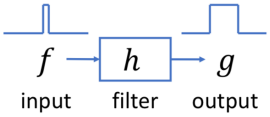
An intuitive way to show the intensity distribution

Image Filtering

1.Filter

In signal processing, filter is a device/process to remove some unwanted components or features from a signal.

Images can be regarded as 2D signal, therefore 2D filters can be used for processing images (smoothing, sharpening, denoising, edge detection ..)



Linear filter if $h(\alpha f_1 + \beta f_2, t) = \alpha h(f_1, t) + \beta h(f_2, t)$

Shift-Invariant filter if $h(t - \Delta t) = h(f(t - \Delta t), t)$ response of filter not depends on t, h=h(x(t))

—Many filters in use are linear shift-invariant filters, for which the output g can be described by convolution of input f and the filters impulse response h.

2.Convolution & Cross-Correlation

convolution of a signal f and a filter impulse response h is $g(t) = f(t) * h(t) = \int_{-\infty}^{\infty} f(\tau) h(t - \tau) d\tau$

discrete form $g[n] = f[n] * h[n] = \sum_{m=-\infty}^{\infty} f[n] h[n - m] = \sum_{m=-\infty}^{\infty} f[n - m] h[m]$

h[n-m]: impulse response filter, convolution kernel

the impulse response h[n] describes what impact a signal f[0] will have on the output side after n steps

convolution of f and h is equivalent to convolution of h and f

properties: commutativity f*h=h*f associativity f*(g*h)=(f*g)*h distributivity f*(g+h)=f*g+f*h

differentiation $\frac{d}{dx}(f*g)=\frac{df}{dx}*g=f*\frac{dg}{dx}$

cross-correlation $g[n]=f[n]*h[n]=\sum_{m=-\infty}^{\infty} f[m]h[n+m]$

3.2D convolution

$$f[m,n]*h[m,n]=\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} f[i,j]h[m-i,n-j]=\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} f[m-i,n-j]h[i,j]$$

f: image h: convolution kernel

$$f[m,n]*h[m,n]=\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} f[m-i,n-j]h[i,j]=\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} f[m+i,n+j]h[-i,-j]$$

h is flipped kernel

If h is symmetric h[i, j]=h[-i, -j], then convolution is equivalent to cross-correlation.

Perform 2D Convolution

Step 1: flip the kernel

Step 2: multiply the image and kernel pixel by pixel

Step 3: sum over the support region of the kernel

The output image is smaller than the input.

To deal with boundary pixels, Padding by constant value (e.g 0), by mirroring values…

4.Moving Average $\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\xrightarrow{normalize}\begin{bmatrix}1/9&1/9&1/9\\1/9&1/9&1/9\\1/9&1/9&1/9\end{bmatrix}$

Computation Complexity:

image size N×N, kernel size K×K

At each pixel, K^2 multiplications and K^2-1 summations

Do this for N^2 pixels. In total, N^2K^2 multiplications and $N^2(K^2-1)$ summations

The complexity is $O(N^2K^2)$

To accelerate it, use Separable filters

Moving average filter removes high-frequency signal and results in a smooth but blurry image.

5.Separable Filter

associativity property of convolution f*(g*h)=(f*g)*h

If a big filter can be separated as the convolution of two small filters, such as g*h, then we can first convolve the image f with g, then with h.

$$\begin{bmatrix}1/9&1/9&1/9\\1/9&1/9&1/9\\1/9&1/9&1/9\end{bmatrix}=\begin{bmatrix}1/9\\1/9\\1/9\end{bmatrix}*\begin{bmatrix}1/9&1/9&1/9\end{bmatrix}$$

Computation Complexity:

two kernels: first one 1×K, second one K×1

At each pixel, K multiplications then K-1 summations

Do this for N^2 pixels

Do this twice for two kernels

That is $2N^2K$ multiplications and $2N^2(K-1)$ summations

The complexity is $O(N^2K)$, versus the original complexity $O(N^2K^2)$

It makes a difference if K is large

6.Hand-crafted Filter

Identity Filter $\begin{bmatrix} & 1 & \\ & & \end{bmatrix}$

Low-pass/Smoothing Filter (moving average, Gaussian)

Gaussian Filter

Kernel: 2D Gaussian Distribution $h(i,j)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$

support is infinite, but we may ignore small values outside [-kσ, kσ]

separable filter h(i, j)=h(i)*h(j) with $h(i)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{i^2}{2\sigma^2}\right)$

High-pass/Sharpening Filter

design 1 $\begin{bmatrix} & 1 & \\ & & \end{bmatrix}+\left(\begin{bmatrix} & 1 & \\ & & \end{bmatrix}-\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}\right)=\begin{bmatrix} -1/9 & -1/9 & -1/9 \\ -1/9 & 1.9 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{bmatrix}$

design 2 $\begin{bmatrix} & 1 & \\ & & \end{bmatrix}+\begin{bmatrix} -1/8 & -1/8 & -1/8 \\ -1/8 & 1 & -1/8 \\ -1/8 & -1/8 & -1/8 \end{bmatrix}=\begin{bmatrix} -1/8 & -1/8 & -1/8 \\ -1/8 & 2 & -1/8 \\ -1/8 & -1/8 & -1/8 \end{bmatrix}$

Denoising Filter

Median Filter

cannot be described by convolution, a non-linear filter

often used for denoising

operations: move the sliding window, replace the center pixel using the median value in the window

7.Learnt Filter: learnt from data

we can manually design filters for various applications (hand-crafted)

Low-pass filters for smoothing

High-pass filters for sharpening

Denoising filters

The filters can also be learned from data, if we have a dataset of input and output images. (will learn in deep learning)

Edge Detection

1.Edge: lines where image brightness changes sharply and has discontinuities

due to color discontinuities, depth discontinuity, surface normal discontinuity etc.

Edges capture important properties of what we see in this world.

Edges are important low-level features for both human vision system and computer vision to analyze images

An image can be regarded as function of pixel position.

Derivatives characterize the discontinuities of a function, which can help us detect edges.

2.

$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x-h)}{2h}$ finite difference for first derivative (central) $f'[x]=\frac{f[x+1]-f[x-1]}{2}$

Prewitt Filter $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}*\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}=\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ (along x-axis/horizontal direction)

Prewitt = moving average for smoothing * finite difference

implement central difference using convolution kernel

Sobel Filter $\begin{bmatrix}1\\2\\1\end{bmatrix}*\begin{bmatrix}1&0&-1\end{bmatrix}=\begin{bmatrix}1&0&-1\\2&0&-2\\1&0&-1\end{bmatrix}$ = smoothing * finite difference (along x-axis/horizontal direction)

$\begin{bmatrix}1\\0\\-1\end{bmatrix}*\begin{bmatrix}1&2&1\end{bmatrix}=\begin{bmatrix}1&2&1\\0&0&0\\-1&-2&-1\end{bmatrix}$ (along y-axis/vertical direction)

Image Gradient

Compute the derivatives along x-axis and y-axis $g_x = f * h_x \quad g_y = f * h_y$

Compute the magnitude of gradient $g = \sqrt{g_x^2 + g_y^2}$

Compute the direction/angle of gradient $\theta = \arctan 2\left(g_y, g_x\right)$

Smoothing

Derivatives are sensitive to noise

smoothing kernel can help suppress the noise

3.Derivative of Gaussian

Gaussian kernels are commonly used for smoothing $h(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{x^2}{2\sigma^2}\right)$

Differentiation of convolution $\frac{d}{dx}(f*g) = \frac{df}{dx}*g = f*\frac{dg}{dx}$

Derivative of Gaussian $\frac{d}{dx}(f*h) = \frac{df}{dx}*h = f*\frac{dh}{dx} = f*\frac{-x}{\sqrt{2\pi}\sigma^3}\exp\left(-\frac{x^2}{2\sigma^2}\right)$

2D Gaussian (separable filter)

Parameter σ in derivative of Gaussian

Large σ value results smoother derivative, which removes noise but blurs edge

different σ values finds edges at different scales

4.Laplacian filter $\begin{bmatrix}1&-2&1\end{bmatrix}*\begin{bmatrix}1\\-2\\1\end{bmatrix}=\begin{bmatrix}0&1&0\\1&-4&1\\0&1&0\end{bmatrix}$

$f''(x) = \lim_{h\to 0}\frac{f(x+h)-2f(x)+f(x-h)}{h^2}$

Laplacian is the sum of second derivatives for 2D images $\Delta f = \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Laplacian of Gaussian

The second derivative is more sensitive to noise than the first derivative

We can also smooth the image using a Gaussian kernel, which results in a Laplacian of Gaussian filter

LOG kernel $\Delta(f*h) = f*\left(\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2}\right) = f*\frac{1}{\pi\sigma^4}\left(\frac{x^2+y^2}{2\sigma^2}-1\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$

with 2D Gaussian Distribution $h(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$

LOG **flipped/negative** is like Mexican hat wavelet(Marr's wavelet)

used in wavelet analysis to extract features from signals

5.Canny Edge Detection

Given filter magnitude maps, where are edges? Can we generate a binary edge map?

Canny proposed an edge detection algorithm

- perform gaussian filtering to suppress noise
- calculate the gradient magnitude and direction (after Gaussian filtering/smoothing)
  - The effect of smoothing can be adjusted by parameter σ in the Gaussian kernel
  - Calculation of image gradient can be
    - in one step by using DOG
    - in two separate steps by Gaussian filter + Prewitt or Sobel filter
- apply non-maximum suppression (NMS) to get a single response for each edge
  - Edge occurs where the gradient magnitude reaches the maximum
  - Check whether the pixel is a local maximum along gradient direction
    - It may require checking interpolated pixels
  - Alternatively, round the gradient directions into 8 possible angles (0°, 45°, 90°, 135°, 180°..315°)
    (horizontal, vertical, diagonal, anti-diagonal)
  - Suppression $M(x,y)=\begin{cases} M(x,y) & local\ \max \\ 0 & else \end{cases}$
- perform hysteresis thresholding to find potential edges
  - The most common method to convert an intensity image to a binary image

$$g(x,y)=\begin{cases} 1,\ if\ f(x,y)\geq t \\ 0,\ otherwise \end{cases}$$

  - Hysteresis Thresholding
    - define two thresholds t_{low} and t_{high} for edge detection
    - If a pixel gradient magnitude is larger than t_{high}, it is accepted as an edge.
    - If a pixel gradient magnitude is lower than t_{low}, it is rejected.
    - If in between, it may be an edge.
      - consider its neighbouring pixels, it will be accepted if it is connected to an edge.

Effect of Gaussian Kernel parameter

- Choice of σ depends on desired behaviour.
- A large σ detects large scale edges; A small σ detects fine features.

Canny proposed performance criteria for an edge detector

- good detection: there should be a low probability of failing to mark real edge points, and low probability of falsely marking non-edge points
  - Gaussian smoothing to suppress noise (reduce false positives)
  - Hysteresis thresholding to find weak edges (reduce false negatives)
- good localization: the points marked as edge points by the operator should be as close as possible to the center of the true edge
- single response: only one response to a single edge
  - Utilize both gradient magnitude and direction, apply NMS

6.Data-driven edge detection

- learning the mapping from image to edge directly from data
- learning an end-to-end network from image to edge map
- It utilizes information at multiple scales (Canny edge detector only works on one scale, controlled by σ)
- It requires ground truth annotation of edges

7.Hough Transform

- Get a parametric representation of edges
- Hough Transform is a transform from image space to parameter space,
  - which output a parametric model, given the input edge points.
- Basic idea—each edge point votes for possible models in the parameter space

Line Parameterization

- Slope intercept form $y = mx + b$ (m slope, b y-axis intercept)

  In practice, the parameter space is divided into 2D bins.
  Each edge point increments the vote by 1 in one of the bins
  Problem with slop intercept form:

  parameter space is too large, m∈(-∞, ∞) b∈(-∞, ∞), we need a lot of bins

- Double intercept form $\frac{x}{a} + \frac{y}{b} = 1$ (a x-axis intercept, b y-axis intercept)
- Normal form $x\cos\theta + y\sin\theta = \rho$ (θ angle, ϱ distance from origin)

  ϱ∈(-∞, ∞) θ∈[0, π), we save a lot of bins

Circle Parameterization $(x-a)^2 + (y-b)^2 = r^2$

The image space (x, y) is transformed to the parameter space (a, b, r)
This is a very large search space, a lot of bins

Interest Point Detection

Feature Description

Image Classification

—digit classification
—image classification (ImageNet Large Scale Visual Recognition Challenge ILSVRC, 1.2million images, 1000 classes)

input image → *input image* —*feature extraction*→ *feature representation* —*classifer*→ *class score output label*

feature extraction: transform input images into low-dimensional vectors so that they can be easily compared or matched

classifer: trained with a dataset consisting of images and labels (or even without labels)

Normally, we split the dataset into two parts

Training set (for training the classifier)
Test set (for evaluating the performance of the classifier on data it has never seen)

Supervised Learning—the labels are available for the training data
Unsupervised Learning—the labels are NOT available for the training data (clustering is often performed for classification)
Semi-Supervised Learning—the labels are available for part of the training data

1.Dataset:

MNIST Modified National Institute of Standards of Technology
Handwritten digit recognition
60000 training samples, 10000 test samples, each sample is a 28×28 pixel image

2.Pre-Processing

- detect where the digit is in a much bigger image
- normalize the size of each digit to 28×28
- normalize the location, place the mass center of the digit in the center of the image
- sometimes, slant correct may also be performed, which shifts each row in the image so that the principal axis become vertical

MNIST provides 2 versions of datasets—regular dataset(centred), deslanted dataset(centred+deslanted)

3.Feature extraction

- Hand-crafted (pixel intensities, principal component analysis)
- Learnt

4.Linear Classifier

①**K nearest neighbours** (KNN)

non-parametric classifier
K=1, each test data points is assigned the class of its nearest neighbour

K>1, compute K nearest neighbours and the test data point is assigned the class given by majority vote
Distance Metric

Euclidean distance $D(x,y)=\sqrt{(x_1-y_1)^2+..+(x_n-y_n)^2}$ (L²-norm)

- If each dimension of the feature x has quite different scales, it is better to normalize the feature vector. (normalize each dimension to a Gaussian distribution N(0,1))

Cosine distance $D(x,y)=\frac{x\cdot y}{||x||||y||}=\frac{x_1y_1+..+x_ny_n}{\sqrt{\sum_{i=1}^{n}x_i^2}\sqrt{\sum_{i=1}^{n}y_i^2}}$

Manhattan distance $D(x,y)=\sum_{i=1}^{n}|x_i-y_i|$ (L¹-norm) General Lᵖ-norm $D_p(x,y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{1/p}$

Features: 28×28 pixel intensity values = 784-dimension vector
Pros: no training step at all
simple but effective (5% error rate on MNIST, 2.4% error rate on MNIST deslanted version)
non-linear decision boundary
multi-class classification
Cons: storage and search are expensive
all training data need to be stored and searched (if a training set of millions of images?)
Computational complexity (N training images, M test images)
there is no training time
At test time, the computational cost to classify M test images is O(MN)
Maybe we want something opposite — we are fine with slow training, but want it to be fast at test time
For pre-processed digit images, maybe it is fine to use 28×28 pixel intensity values as the feature vector
But for general images, simply using the Euclidean distance between the full images may not be a good choice.
(not invariant to scale, rotation etc)

Hyper-parameters
what is the best value of K to use for classification algorithm? what is the best distance metric to use?
It is problem-independent
It is often just trial and error. Try a lot of parameters, train your model and see which ones work the best.

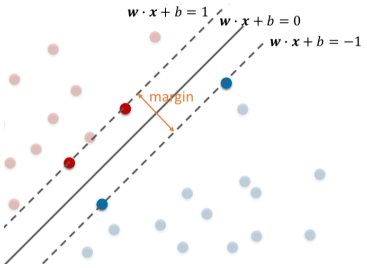
- Can we tune the hyper-parameters to see which ones work best on the test set?
- Not a very good practice. The test set is for evaluating the model performance on unseen data. We had better only use it once at the end.
- For challenges and competitions, we will not get the ground truth for the test set.

Cross Validation
split the training set into 2 parts—training set and validation set, then tune hyper-parameters on the validation set
Once you know the best parameters, train your model on the full training set
and evaluate its performance on the test for just once
you can cycle through each fold, using it as a validation set
For example, we can train the model for 5 times and calculate the average performance for some parameter values
However, if it is expensive to train model (deep neural networks, a full day to train one model), just do this once.
Training set—train the model
Validation set—decide which type of model and which hyper-parameters are the best
Test set—get a final estimate of model performance.
(we expect the test performance to be worse than the validation performance)

If you are participating in a challenge, finally, train your model on the full training set and submit your classification results onto the challenge/competition websites, which will perform the evaluation.

②Support vector machine

Linear Classifier



For a 2D case (input data is 2D), the line has the form $w_1x_1+w_2x_2+b=0$
The rule of the linear classifier is to assign a class c to data x:

$$c = \begin{cases} +1, \ if \ w_1 x_1 + w_2 x_2 + b \geq 0 \\ -1, \quad otherwise \end{cases}$$

For KNN, we need to store all the training data.

For linear classifier, once we know w and b, we can discard the data.

For more general cases, the linear model is given by $w \bullet x + b = 0$

w: weights, the normal x: data b: bias

The rule of the linear classifier is to assign a class c to data x: $c = \begin{cases} +1, \ if \ w \bullet x + b \geq 0 \\ -1, \ otherwise \end{cases}$

- To train the classifier, what we need is to find the parameters w and b, which determines the decision boundary/hyperplane. (a line that are far from both classes of points—maximum margin hyperplane)
- To determine the maximum-margin hyperplane, we only need the most inner points (support vectors)
- we would like the support vectors to fulfill the equations $w \bullet x + b = +1 \ or \ -1$ so that all the other points are easily classified. we would also like to maximize the margin between the dashed lines.

Maximum Margin

- margin between $w \bullet x + b = +1$ and $w \bullet x + b = -1$
- Let x_1 be a point on the plane $w \bullet x + b = -1$ so that $w \bullet x_1 + b = -1$
- Move x_1 along the normal direction n for the distance of the margin m, it should arrive at the point x_2 on the other plane $w \bullet x + b = 1$
- We have $w \bullet (x_1 + mn) + b = 1$, so $mw \bullet n = 2$
- Known that the normal $n = w / ||w||$. The margin $m = 2 / ||w||$
- $\max_{w,b} \frac{2}{||w||}$ subject to $w \bullet \underset{data}{x_i} + b \begin{matrix} \geq +1 \ if \ y_i = +1 \\ \leq -1 \ if \ \underset{class}{y_i} = -1 \end{matrix}$ for i=1..N ≡ $\min_{w,b} ||w||^2$ s.t. $y_i (w \bullet x_i + b) \geq 1$ (i=1..N)

This is a quadratic optimization problem subject to linear constraints

Possible Mistakes & Slack Variables

- Now the points are no longer linearly separated.
- Introduce the slack variable ξi≥0 for each data point
- $\min_{w,b} ||w||^2 + C \sum_{i=1}^{N} \xi_i$ (minimize the distance) s.t. $y_i (w \bullet x_i + b) \geq 1 - \xi_i$ (i=1..N) (distance from the margin)
- If all points can be correctly classified with $y_i (w \bullet x_i + b) \geq 1$, then ξi=0. The loss function still the same.
- Only when some points are inside the margin or wrongly classified, we will have a positive ξi in the loss
- If we allow ξi to be sufficiently large, we can accommodate all the wrongly classified points
- C is a regularization term (hyper-parameter)

  Small C allows constraints to be easily fulfilled (soft margin)

  mistake is tolerated in order to get a larger margin

  large C makes constraints hard to be ignored (hard margin)

find a separate plane with the largest margin, solve the quadratic programming (dual) problem

Multi-class classification

- 1 vs rest

  train a classifier for each of the 10 digits,

  during test, apply all the 10 classifiers to the test data

  the classifier which produces the highest response will determine the result $c = \arg\max_{k=1..K} f_k(x)$
- 1 vs 1

  train a classifier between each pairs of digits. (For K classes, will get ½K(K-1) classifiers)

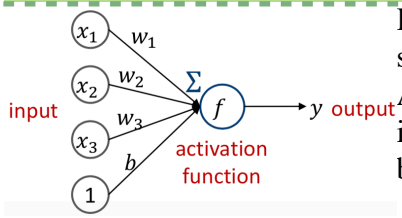
  during test, each classifier will vote for a digit

  count the vote for each digits and perform majority voting

③Neural Network: Multi Layer Perceptron

④Neural Network: Convolutional Neural Network

Neural Network: Multi Layer Perceptron



Biology: neurons are inter-connected,
signal(electrical, chemical) flows from input at dendrites to output at axon terminals
Artificial Neural Networks:
inspired by biology,
but not the model of how brain works, not exact model how neuron works

Neuron: a computational unit that takes an input, applied an activation function and generates an output
A neural network is formed by putting many neurons into connection, where the output of a neuron can be the input to another. It often consists of several layers of neurons.

1.Multi-layer perceptron (MLP), which is a fully connected multi-layer network.

Forward Propagation—process that signal flows from input layer to output layer

For layer i, $z^{i+1} = W^i a^i + b^i$ and $a^{i+1} = f\left(z^{i+1}\right)$ where f is activation function applied element-wise

Loss function (least square error)

- because of the non-linear activation function f, it is not a quadratic programming problem.
- but it is differentiable, we can use gradient descent to minimize the cost function

$$W^{(k+1)} = W^{(k)} - \eta \nabla_W J\left(W^{(k)}, b^{(k)}\right) \text{ and } b^{(k+1)} = b^{(k)} - \eta \nabla_b J\left(W^{(k)}, b^{(k)}\right)$$

Back Propagation—to calculate the gradient of J(W, b)

- It is an algorithm to propagate the error (discrepancy between prediction and ground truth) from the last layer backwards to previous layers
- It is used to evaluate the gradient of the loss function with regard to the network parameters W and b
- It is based on the chain rule in differentiation
  - To calculate the derivatives at layer i, we only need the derivatives at layer i+1.

2.Stochastic Gradient Descent

- Initialize the network parameters W and b
- Set the batch size B, number of iterations and learning rate α
- For each iteration
  - randomly select a mini-batch of B training samples, calculate batch gradient and update parameters

Epoch: one pass through the whole training set

If there are N training samples and use B batch size, then it takes N/B iterations to complete on epoch

Hyper-Parameters: batch size B, learning rate α

- For SGD, a small batch size means we can evaluate the gradient quicker.
  If batch size is too small, the gradient may become sensitive to a single training sample
  If batch size is too large, computation will become expensive and use more memory on GPU.
- Learning rate/step length/step size need a good learning rate
  With a low learning rate, the optimization may be slow.
  With a high learning rate, the optimization is faster but may bounce chaotically and not arrive at good local minima
  With a very high learning rate, the optimization may diverge

Problems in optimization

- Vanishing Gradient: gradient becomes vanishingly small, preventing the weights from changing their values
  - e.g. sigmoid function—when $f(z)$ saturates at 0 or 1 and f'(z) becomes almost 0, $\delta^{(l)}$ becomes negligible.
  - Activation function
    - Rectified Linear Unit (discontinuity at 0 but has sub-gradient), outperform the sigmoid function
      - gradient will not vanish when z is very large, but gradient becomes 0 when z becomes negative, which may still lead to vanishing gradient problem
    - Leaky ReLU f(z)=0.01z, z<0; z, z>0
    - Parametric ReLU f(z)=az, z<0; z, z>0 (learning parameter a in training)
    - Exponential Linear unit f(z)=a(e^z − 1), z<0; z, z≥0
  - they allow a small gradient when z is negative and may outperform ReLU, depending on applications
- Exploding Gradient: gradient becomes very large, preventing the algorithm from converging
  - gradient clipping (clip by value, clip by norm)

3.Loss function

Neural networks
Build a network
Input layer—Images, labels (one-hot encoding)
Hidden layers
Output layer—Regression: mean squared error Classification: cross-entropy loss
Train the network—Calculate the gradient using back propagation & Stochastic gradient descent
Evaluate the network—Regression/classification accuracy

Neural Network: Convolutional Neural Network

Output Size
Input Shape $X_{in} \times X_{in}$
Kernel Size k×k
Padding p×p
Stride s×s
Dilation d×d
Output Shape $k_d = k + (k-1) * (d-1)$ $X_{out} = \left\lfloor \frac{X_{in} + 2p - k_d}{s} \right\rfloor + 1$

Convolutional Neural Networks Examples

Object Recognition & Detection

1.Basic Idea:
Already built a NN that can perform classification for a given image
Move a sliding window and apply the classification network to different regions of a image for detection.
At each sliding window, perform 2 tasks
①classification (whether this is a object or not)
②localization (bounding box coordinates), a regression problem
The sliding window provides rough localization. Using CNN features, we can get refined localization.
2.Two-stage detection
①region proposal
②classification and localization

	Stage 1: region proposal	Stage 2: classification & localization	
R-CNN	Selective Search	SVM for classification, linear regression for localization, both using CNN features	RCNN = Regions with CNN features
Fast R-CNN	Selective Search	CNN	
Faster R-CNN	CNN	CNN	
Mask R-CNN	CNN	CNN	CNN for mask segmentation

Mask R-CNN = Faster R-CNN + mask branch for segmentation
three branches for the pooled ROI (classification, localization, segmentation)

3.One-stage object detection method
YOLO
Single Shot Multibox Detector SSD

4.The performance of object detection methods also depends on the backbone network/feature extractors, which provides the convolutional feature map. (choices: AlexNet, VGG, ResNet)
The performance also depends on image resolution.

Image Segmentation (semantic segmentation)

Image segmentation is a process of partitioning an image into multiple regions, each region consisting of pixels with similar properties (intensity, color, texture) or semantics.

image segmentation—each class gets a unique label

instance segmentation—each instance gets a unique label

1.Clustering(unsupervised learning)

Thresholding (simplest method for segmentation)

It converts a grayscale image into a binary label map

At each pixel x, the label is defined by $f(x)=\begin{cases} 1, \ if \ I(x) \geq threshold \\ 0, \qquad otherwise \end{cases}$

No training data is involved. The only parameter is the threshold.

It assumes that pixel intensities can be grouped into two clusters and a threshold can separate the two.

General cases, more than 2 clusters.

How to define these clusters and determine which cluster each pixel belongs to?

K-means clustering provides a simple way for cluster analysis,

by iteratively estimating cluster parameters and pixel associations.

- Represent each cluster by its center. Each data point (pixel intensity) is associated to the nearest cluster center
- The optimal cluster centers are those that minimize the intra-class variance, defined as the sum of squared difference between each data point and its associated cluster center $\min \sum_{k=1}^{K} \sum_{x \in C_k} (x-\mu_k)^2$
- $\min \sum_{k=1}^{K} \sum_{x} \delta_{x,k} (x-\mu_k)^2$

  $\delta_{x,k}$: whether x is assigned to cluster k (2 unknowns—membership δ & cluster center μ)

The feature can be more than just intensities (color similarity, position+color similarity, other features)

Unsupervised method, no training data needed. Easy to implement and fast. But not provide any semantic information.

2.Convolutional neural network(supervised learning method)

Semantic Segmentation ≈ pixel-wise classification

Moving a sliding window and applying the classification network to each pixel of a image for segmentation

It may be too expensive to apply the network to so many sliding windows at all the pixels

The output is a probability vector for a whole region.

Can we develop a network which outputs a pixel-wise probability?

Upsampling

Transposed Convolution

Convolution as a matrix operation

3.Mask R-CNN

Motion Estimation (Video Analysis and Motion Tracking)

Previously, we mainly look at understanding static images.

Motion information is useful, for action recognition and object tracking.

1.Video

A video is an 2D-t image sequence captured over time.

The image date is a function of space(x, y) and time t

2.Optic Flow

Optic flow is a method for estimating the motion between two frames of images

It outputs a flow field, which has a displacement vector associated with each pixel of the image.

Two basic Assumption:

- brightness constancy
  the brightness of an image object does not change over time
- small displacement
  the displacement between two consecutive frames is small

Three types of methods:

- local method—Lucas-Kanade Method
- global method—Horn-Schunck Method
- Learning-based method

3.Lucas-Kanade Method

Based on the brightness constancy assumption $I(x+u,y+v,t+1)=I(x,y,t)$

I intensity (x, y, t) spatial and temporal coordinates (u, v) displacement

First-order Taylor expansion $I(x+u,y+v,t+1)\approx I(x,y,t)+\frac{\partial I}{\partial x}u+\frac{\partial I}{\partial y}v+\frac{\partial I}{\partial t}$

Optical Flow Constraint Equation $\frac{\partial I}{\partial x}u+\frac{\partial I}{\partial y}v+\frac{\partial I}{\partial t}=I_x u+I_y v+I_t=0$ 1 equation, 2 unknowns(cannot solve)

I_x, I_y spatial gradient, I_t temporal gradient

Another assumption—flow/displacement is constant with a small neighbourhood

At each pixel p, optic flow constraint equation $\begin{bmatrix} I_x(p) & I_y(p) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}=-I_t(p)$

For a small neighbourhood (3×3 window), a system of linear equations Ax=b

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_N) & I_y(p_N) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}=-\begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_N) \end{bmatrix}$$ overdetermined (more equations than unknowns)

Least Square Method $x=\arg\min_x ||Ax-b||^2$ Solution $x=(A^TA)^{-1}A^Tb$

$$A^TA=\begin{bmatrix} \sum_p I_x^2 & \sum_p I_xI_y \\ \sum_p I_xI_y & \sum_p I_y^2 \end{bmatrix}=\sum_p\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \qquad A^Tb=-\begin{bmatrix} \sum_p I_xI_t \\ \sum_p I_yI_t \end{bmatrix}$$

Limitations

•Many modern computer vision algorithms are based on deep neural networks, which are very powerful in learning features or representations.

•However, these approaches might have certain limitations:

- Data hungry.
- Maybe over-parameterized.
- Sometimes may not generalize well, especially on data from an unseen domain.
- May not be well integrated with prior knowledge.
- ...

Tutorial & Coursework

CW1:

①Moving Average Filter

large windows make the results more blurry

increased blurring due to a large kernel, more noise is suppressed with a larger kernel
②Edge Detection (Sobel Filter, Gaussian+Sobel filters, 2D & 1D)
Gaussian smoothing suppresses noise in gradient magnitude map
separable filtering accelerates image filtering or leads to speed-up
③Laplacian Filter (3×3 Laplacian, LoG)
Gaussian smoothing suppresses noise, which benefits Laplacian filtering results

CW2:
If the dataset is not balanced, the machine learning model will be biased towards the class with more samples.
MLP

batch normalization
layers: relu activation
last layer: softmax activation
dropout:

epoch
batch

confusion matrix: true label × predicted label

Question 1.1: image transformation (scaling, translation, rotation)
A translation operation. It translates the image along the x-axis by 10 pixels and along the y-axis by 5 pixels. The pixel (10, 5) in the new image corresponds to the pixel (0, 0) in the old image.
A scaling operation. It zooms the image by 5-fold. The pixel (5, 5) in the new image corresponds to the pixel (1, 1) in the old image.

Question 1.3: Softmax function $p_i = \frac{\exp(c_i)}{\sum_k \exp(c_k)}$

- if i=j then $\frac{\partial p_i}{\partial c_j} = \frac{\exp(c_i)\sum_k \exp(c_k) - \exp(c_i)\exp(c_j)}{\left(\sum_k \exp(c_k)\right)^2} = \frac{\exp(c_i)}{\sum_k \exp(c_k)} \frac{\sum_k \exp(c_k) - \exp(c_j)}{\sum_k \exp(c_k)} = p_i(1 - p_j)$
- if i≠j then $\frac{\partial p_i}{\partial c_j} = \frac{-\exp(c_i)\exp(c_j)}{\left(\sum_k \exp(c_k)\right)^2} = \frac{\exp(c_i)}{\sum_k \exp(c_k)} \frac{-\exp(c_j)}{\sum_k \exp(c_k)} = p_i(-p_j)$

Using Kronecker Delta $\delta_{ij} = \begin{cases} 1, if\ i = j \\ 0, if\ i \neq j \end{cases}$ Derivative $\frac{\partial p_i}{\partial c_j} = p_i(\delta_{ij} - p_j)$

256=2^8, 1 byte=8 bits, 1024 byte=1MB

Question 2.1: Gradient Magnitude & Gradient Direction (Prewitt filter, Sobel Filter)
Note: flip the kernel !!! $g_x = f * h_x$ $g_y = f * h_y$ $g = \sqrt{g_x^2 + g_y^2}$ $\arctan 2(g_y, g_x)$

Question 2.3: 2D Gaussian Filter (infinite support) $h(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$

$$\frac{\partial h}{\partial x} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)\left(-\frac{2x}{2\sigma^2}\right) = \frac{-x}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

After performing the convolution, we will see that it is equivalent to the 2D Gaussian filter. This indicates that the 2D Gaussian filter is separable as two small 1D filters.

Question 2.4: noise in combined image $e = Y - I = \frac{1}{N}\sum n_i$

Question 3.1: Hough transform y=mx+b → b=y-mx (votes)

Question 3.1: min $E(\beta)=(Y-X\beta)^T(Y-X\beta)=Y^TY-2\beta^TX^TY+\beta^TX^TX\beta$

$\partial E/\partial\beta=-2X^TY+2X^TX\beta=0$ $\beta=(X^TX)^{-1}X^TY$ and $\partial^2E/\partial\beta^2=2X^TX$ is positive definite, so it is a local minimum

Question 4:

Harris detector for interest point detection $M=\sum w(x,y)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$ $Mu=\lambda u$ and $(M-\lambda I)u=0$

Harris detector response $R=\lambda_1\lambda_2-k(\lambda_1+\lambda_2)^2$

SIFT (use scale-adapted Harris detector) to compare Harris detector response between different scales

Laplacian operator $\nabla^2L=\frac{\partial^2L}{\partial x^2}+\frac{\partial^2L}{\partial y^2}$

Question 5.1: sigmoid/logistic function $f(x)=\frac{1}{1+e^{-x}}$ $f'(x)=\frac{-e^{-x}(-1)}{(1+e^{-x})^2}=\frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}}=f(x)[1-f(x)]$

The sigmoid function suffers from the vanishing gradient problem. When $f(x)$ saturates at either 0 or 1, its derivative $f'(x)$ becomes nearly 0.

Use Rectified linear unit $ReLU=\begin{cases}0,x<0\\x,x\geq0\end{cases}$ to address this problem, its sub-gradient $ReLU'=\begin{cases}0,x<0\\1,x\geq0\end{cases}$

Question 5.2: $f(x)=x\sigma(\beta x)=\frac{x}{1+e^{-\beta x}}$

When $\beta\to0$, $f(x)=\frac{x}{1+e^0}=\frac{x}{2}$ When $\beta\to\infty$, if $x\geq0$, $f(x)=\frac{x}{1+e^{-\infty}}=x$; if $x<0$, $f(x)=\frac{x}{1+e^{\infty}}=0$

$$f'(x)=\sigma(\beta x)+x\frac{\partial\sigma(\beta x)}{\partial x}=\sigma(\beta x)+x\frac{\partial\sigma(\beta x)}{\partial\beta x}\beta=\sigma(\beta x)+x\sigma(\beta x)[1-\sigma(\beta x)]\beta=\beta x\sigma(\beta x)+\sigma(\beta x)[1-\beta x\sigma(\beta x)]=\beta f(x)+\sigma(\beta x)[1-\beta f(x)]$$

Question 6:

Aspect Ratio = Width / Height

TP: # of true positives—item correctly detected as belonging to the positive class

FP: # of false positives—item wrongly detected

FN: # of false negatives—item missed in detection but belonging to the positive class

Precision = TP/(TP+FP) Recall = TP/(TP+FN)

average precision: sample the precision-recall curve at different recall values and average

Question 7.1: CNN

Data Shape: $k_d=k+(k-1)*(d-1)$ $X_{i+1}=\left\lfloor\frac{X_i+2p-k_d}{s}\right\rfloor+1$

The **receptive field** is defined as the region in the input space that a particular CNN's feature is looking at (be affected by). A receptive field of a feature can be described by its **center location** and its **size**.

To calculate the receptive field size r, we will need the distance between two adjacent features
j (known as the jump). Initially $j_0=1$ and $r_0=1$. Then calculate the receptive field using the following equations:
$j_{i+1}=j_i*s$ and $r_{i+1}=r_i+(k-1)*j_i$, where s is the stride size and k is the kernel (filter) size.

The latest GPU card enables the use of half precision floating point, i.e. 2 bytes per number. Discuss the advantages and disadvantages using half precision for training a neural network.
Half precision will reduce the memory cost and accelerate training speed on GPUs. However, it may sacrifice the numerical accuracy. In practice, mixed precision may be adopted, in which both half precision and single precision are used.

Question 7.2: Motion
optic flow constraint $I_xu+I_yv+I_t=0$
assume flow is constant within 3×3 neighbourhood — (u, v) in the window are the same